U.P.B. Sci. Bull., Series C, Vol. 81, Iss. 4, 2019 ISSN 2286-3540

A RECOMMENDATION ALGORITHM BASED ON THE HYBRID MODEL

Bin LI[1], Ning MA[2], Ninghui LI[3], Yuliang GUO[4]

With great growth of information resources on the internet, how to recommend interesting items from mass data to users with different interests and hobbies according to their information characteristics has become an urgent problem to be solved. In this paper, we improved the recommendation algorithms of SVD (singular value decomposition) and KNN (K-nearest neighbor algorithm) by different characterization factors, and then proposed a hybrid algorithm based on improved algorithms. The test results on MovieLens dataset show that the root mean squared error (RMSE) of scoring prediction by using the hybrid recommendation algorithm reduces greatly and prediction accuracy of the recommendation algorithm also increases significantly.

Keywords: recommendation algorithm, singular value decomposition, K-nearest neighbor algorithm, hybrid recommendation

1. Introduction

With the rapid development of WEB 2.0 and constant expansion of network size, information resources on the internet have greatly increased. People have to spend a lot of time searching for needed information. Compared with the era with deficient information before the emergence of the internet, how to search for needed information from such mass information data nowadays has become an important research topic. Users expect to have a system (or a website) serving as a shopping assistant to help them choose items. The shopping assistant can automatically choose items based on users' interest and then recommend them to users. The system also hopes to recommend different users items and information which users are most interested in.

The appearance of the recommender system solves the aforementioned problems effectively. The system actively recommends interesting information and items to specific users according to their information requirement, interests

[1] A.P., Dept. of Information Engineering, Anhui Open University, China, e-mail: szbinlee@126.com
[2] A.P., Dept. of Information Engineering, Anhui Open University, China, e-mail: maning@ahou.edu.cn
[3] A.P., Dept. of Information Engineering, Anhui Open University, China, e-mail: liningh@ahou.edu.cn
[4] Ass., Dept. of Open Education, Anhui Open University, China, e-mail: guoyul@ahou.edu.cn

and hobbies and personal history, expecting to help users rapidly search for their contents of interest from mass information. The main task of the recommender system is how to find the items that users are probably interested in, and then recommend them to users in a certain form.

2. Related work

The basic idea of recommender system based on contents is to select some items similar to those that users have purchased or are interested in from candidate objects and include them in the recommendation list [1]. Among the modeling methods for item contents, vector space model (VSM) is the most famous one [2]. As early as 1997, Fab system was developed by Balabanovic and Shoham, which expressed the content of the web page mainly by selecting the most important keywords in the web page. At that time, the number of selected keywords was about 100 [3]. In recent years, by using WordNet, Degemmis et al. constructed user profile based on semantics instead of traditional keyword construction method. The configuration file is not only composed of keywords, but also contains the semantic information on users' preferences [4]. Nguyen et al. proposes a method to efficiently provide better Web-page recommendation through semantic-enhancement by integrating the domain and Web usage knowledge of a website [5]. Langer et al. used Apache Lucene as recommendation framework in scholarly-literature recommender system of the reference-management software Docear [6]. Deldjoo et al. proposed a content-based recommender system that encompassed a technique to automatically analyze video contents and to extract a set of representative stylistic features grounded on existing approaches of Applied Media Theory [7].

Collaborative filtering recommendation algorithm is acknowledged as one of the most famous algorithms in the field of recommender system. Group Lens first proposed user-based collaborative filtering algorithm in 1994 [8]. Amazon (www.amazon.com) put forward item-based collaborative filtering algorithm in 2000 [9]. These two algorithms are the most classical in existing collaborative filtering algorithms [10]. In addition, there are many other algorithms based on collaborative filtering. For example, the co-view graph model was used in the recommendation algorithm of YouTube website. Baluja proposed a diffusion algorithm based on this algorithm, which can measure the users' interest in items on the graph [11]. To deal with the disadvantage of the single rating-based approach, multi-criteria collaborative filtering was developed [12]. Adeniyi et al. presented a study of recommendation system based on current user behavior through user's click stream data on the Really Simple Syndication (RSS) reader website [13]. Jian et al. proposed two recommendation models to solve the complete cold start (CCS) and incomplete cold start (ICS) problems for new items, which were based on a framework of tightly coupled CF approach and deep learning neural network [14]. Karabadji et al. proposed to focus mainly on the growing of the large search space of users' profiles and to use an evolutionary multi-objective optimization-based recommendation system to pull up a group of profiles that maximizes both similarity with the active user and diversity between its members [15].

The neighborhood-based algorithm is regarded as the simplest social filtering algorithm. Jamali and Ester modeled the relationship between users' social network and users' preferences for items into a graph by utilizing graph model and then made social recommendations to the users by applying random walk algorithm [16]. Reshma and Pillai proposed a semantic based trust recommendation system which recommended trust companion having high similarities in message sharing [17]. Logesh et al. expressed views on social network data-based recommender systems by considering usage of various recommendation algorithms, functionalities of systems, different types of interfaces, filtering techniques, and artificial intelligence techniques [18]. Yang et al. proposed a method that works to improve the performance of collaborative filtering recommendations by integrating sparse rating data given by users and sparse social trust network among these same users [19].

3. SVD algorithm and its improvement

3.1. SVD algorithm

The basic idea of singular value decomposition (*SVD*) algorithm is to analyze the factors contained in items that can reflect their characteristics according to known information on items and scores of items in the system to thus further analyze users' degree of preference for each factor [12]. Eventually, items are predicted and scored according to the analysis results of these two steps. By utilizing formalized language, the process can be described as: scoring matrix R is composed of $M\times N$ elements so that the element $R[u][i]$ represents the score of the uth user for the ith item. Afterwards, the scoring matrix R is factorized into the matrix P reflecting users' preference level for item factors and the matrix Q describing characterization factors of items. Those factors of the item can be regarded as the categories, themes and so on. The matrix P consists of $M\times F$ elements and the preference level of the uth user for the factor k is expressed as $P[u][k]$ while the matrix Q is composed of $N\times F$ elements and the degree to which the ith item shows the kth factor is expressed as $Q[i][k]$. The relationship of matrix R with matrixes P and Q can be expressed in Equation (1).

$$R = PQ^T \tag{1}$$

Where, Q^T is the transposed matrix of the matrix Q. A larger value of R means a higher interest of users in an item. The scoring matrix R can be factorized into matrix P with users' preference factors and the matrix *Q* with characterization factors of items.

During *SVD*, the matrix factorization model is based on stochastic gradient descent (*SGD*) can effectively solve the problem relating the factorization of a high-dimensional dense matrix [12]. The basic idea of the model is summarized as follows: the score *R* of the user *u* for the product *i* can be expressed by using eigenvector p_u of users and eigenvector q_i of items according to Equation (1), as shown in Equation (2).

$$\hat{R}_{u,i} = p_u q_i^T \tag{2}$$

In Equation (2), p_u and q_i refer to the preference of the *u*th user for characterization factor of items and the degree to which the *i*th item shows corresponding characterization factor, respectively. Moreover, p_u and q_i can be obtained by training optimized loss function, as shown in Equation (3).

$$C(D) = \sum_{(u,i)\in D_T} (R_{u,i} - p_u q_i^T)^2 + \lambda(\|p_u\|^2 + \|q_i\|^2) \tag{3}$$

Where, D_T refers to the training dataset. $R_{u,i}$ denotes the actual score of a product. Parameter $\lambda(\|p_u\|^2 + \|q_i\|^2)$ is used to avoid over-fitting of data. In terms of *SGD* optimization algorithm, partial derivatives of p_u and q_i need to be solved at first, as shown in Equations (4) and (5).

$$\frac{\partial C}{\partial p_u} = -2q_i e_{ui} + 2\lambda p_u \tag{4}$$

$$\frac{\partial C}{\partial q_i} = -2p_u e_{ui} + 2\lambda q_i \tag{5}$$

Here, $e_{u,i}$ denotes the value of error between predicted and actual score of a product, it can be calculated by $R_{u,i} - p_u q_i^T$. Subsequently, by using SGD algorithm, iterative computations of the above equations are updated as follows:

$$p_u = p_u + \alpha(e_{ui} q_i - \lambda p_u) \tag{6}$$

$$q_i = q_i + \alpha(e_{ui} p_u - \lambda q_i) \tag{7}$$

In Equations (6) and (7), α refers to learning rate.

3.2. Improvement of SVD algorithm

In practice, people may sometimes be influenced by other factors besides their own interests and preferences when rating an item. For example, in terms of information of users' personality characteristics, the different personalities will also lead to the difference of scores. Some users will give full marks to the items they like while some other users can only give 80 marks to items they like. A strict user often gives a lower score on the same item than a tolerant user. Similarly, sometimes when a user notices that scores of an item are mostly high, he tends to give a high score on that item. In order to improve the accuracy of the model, it is necessary to add the aforementioned two problems to improve the *SVD* Equation. The improved Equation is shown as follow.

$$R_{u,i} = overallScore + b_u + b_i + p_u q_i^T \tag{8}$$

Where, *overallScore* refers to the average score of all items in a system (or a dataset) while parameters b_u and b_i denote the differences of the rating of the *u*th user and the score of the *i*th item with *overallScore,* respectively. Hence, b_u and b_i both represent the degree of deviation of scores from the average score. Moreover, p_u refers to the preference of the *u*th user for characterization factor of an item while q_i denotes the degree to which the *i*th item shows corresponding characterization factor.

The optimization learning process of parameters by using stochastic gradient descent (*SGD*) algorithm can be divided into the following steps: at first, each parameter is assigned with a certain initial value. Then, the scores of items are predicted by applying these initial parameter values and also the predicted scores of items are compared with known scores of items. Finally, the parameters are constantly amended and adjusted according to the comparison results. From the perspective of formalization, values of parameters are adjusted so that the value of Equation (9) is minimized.

$$\sum_{(u,i)\in\delta} \left\{ (r_{u,i} - overallScore - b_u - b_i - p_u q_i^T)^2 + \lambda(b_u^2 + b_i^2 + \|p_u\|^2 + \|q_i\|^2) \right\} \tag{9}$$

Where, δ refers to all training sample. Moreover, b_u and b_i are initialized as 0 while the initialization of *P* and *Q* is realized by using *0.1*rand(0,1)/sqrt(dimensions)*. The result of the first parentheses indicates the deviation between the current predicted value and the actual value while the equation in the second parenthesis is mainly used to prevent over-fitting. The improved algorithm is called *NSVD* algorithm.

4. K-nearest neighbor algorithm and its improvement

4.1. KNN algorithm

According to the theory of KNN algorithm [13], the process for predicting the rating of the user *u* for the item *j* by employing *KNN* algorithm can be divided into three steps: calculating similarity, selecting *K* nearest neighbors and calculating predicted value.

4.1.1. Similarity calculation based on Pearson correlation

Owing to the similarity calculation can be conducted based on the similarity between users or between items, the calculation based on the similarity of items is conducted, and the corresponding equation is given. It is supposed that a user scores both items *i* and *j*, and the set of the users can be represented by using $U_{i,j}$. Hence, the similarity *sim*(*i, j*)between items *i* and *j* can be expressed as Equation (10) [13].

$$sim(i,\ j) = \frac{\sum_{c \in U_{i,j}} (R_{c,i} - \bar{R}_i)(R_{c,j} - \bar{R}_j)}{\sqrt{\sum_{c \in U_{i,j}} (R_{c,i} - \bar{R}_i)^2} \sqrt{\sum_{c \in U_{i,j}} (R_{c,j} - \bar{R}_j)^2}} \qquad (10)$$

Where, $R_{c,i}$ denotes the score of the user *c* on the item *i while* $\bar{R}_i$ and $\bar{R}_j$ separately refer to the average scores of items *i* and *j*, respectively.

4.1.2. Selection of k nearest neighbors and calculation of predicted value

Among the items scored by the user *u*, *K* items are the highest similarity with the item j are found through similarity calculation and the set of the K items is expressed as *N*(*u, j*). Generally, the value of *K* is an odd number.

The similarity between items can be obtained by using various methods for calculating similarity. By selecting *K* nearest neighbors, *K* adjacent neighbors of an item can be acquired. In subsequent steps, it is necessary to produce corresponding recommendation objects. Before producing recommendation objects, it needs to attain the score of users on items to be recommended through calculation. The process of calculating the score $\hat{R}_{u,j}$ of the user *u* on item *j* is shown in Equation (11).

$$\hat{R}_{u,j} = \frac{\sum_{n \in N(u,j)} sim(n,\ j) R_{u,n}}{\sum_{n \in N(u,j)} sim(n,\ j)} \qquad (11)$$

Where, *sim*(*n, j*) refers to the similarity between items *n* and *j* and $R_{u,n}$ represents the score of the user *u* on the item *n*.

Through the above steps, it is feasible to predict users’ score on non-scored items and then select the *N* items with the highest predicted score as objects recommended to users.

4.2. Improvement of KNN algorithm

4.2.1. Improvement of data sparsity

With the increase of the number of users and items in a recommender system, the data size of the whole system becomes very large while fewer items are shared by two users, which leads to data sparsity. According to Pearson’s correlation equation, it can be seen that if the size of intersection set of two items is much smaller than that of other items, the similarity between the two items shows a low reliability. When the data size becomes very large, it would be a common phenomenon of having certain small intersection sets in the recommender system due to the presence of data sparsity, which greatly weakens the reliability of similarity result. In order to enhance the reliability of the predicted results, it is necessary to compress the similarity according to the size of the intersection set, as shown in Equation (12).

$$\overline{sim(i,j)} = \frac{|U_{i,j}|}{|U_{i,j}| + \gamma} * sim(i,j) \tag{12}$$

Where, $\overline{sim(i,j)}$ refers to the similarity between compressed items i and j and γ denotes the compression coefficient specified by users.

4.2.2. Improvement of global effect

Users or items themselves have many characteristics, which leads to a subtle scoring trend when users score items. For example, some users belong to strict raters and they tend to give items a low score while some other tolerant raters will score a high value on items. At the same time, there will be a similar situation for items. Some items (such as big brand goods) will generally get a high score even if they perform ordinarily while some items will sometimes not attain a high score even if they perform well. In the recommender system, the trends are called global effect (*GE*). The common GE factors are shown in Table 1 [20].

$$R_{u,j}^{(t+1)} = R_{u,j}^{(real)} - \hat{R}_{u,j}^{(t)} \tag{13}$$

Table 1

Global influencing factors and their meanings

Order	GE	Meaning
0	**Overall mean**	Mean score of all items
1	**Item effect**	Tendency of scores on items
2	**User effect**	Tendency of scores of users
3	**User×Time(user)$^{1/2}$**	Time interval between the first scoring of users to the present (based on users)
4	**User×Time(item)$^{1/2}$**	Time interval between the first time when items are scored to the present (based on users)
5	**Item×Time(item)$^{1/2}$**	Time interval between the first time when items are scored to the present (based on items)
6	**Item×Time(user)$^{1/2}$**	Time interval between the first rating of users to the present (based on items)
7	**User×Item average**	Average score of items (based on users)
8	**User×Item support**	Scored times of items (based on users)
9	**Item×User average**	Evaluation scores of users (based on items)
10	**Item×User support**	Scoring times of users (based on items)

The goal of setting *GE* factors is to estimate a specific parameter for the factors. When estimating parameters, only one factor is considered at a time, and the predictive residuals of all the factors obtained above are used as the true score of this estimation. The true score of the (*t*+1)th factor is estimated according to the *t*th factor, as shown in Equation (13).

In Equation (13), $\hat{R}_{u,j}^{(t)}$ refers to the sum of predicted scores of the first *t* GE factors of the item *j* by the user *u* and $R_{u,j}^{(real)}$ denotes the true score of the user *u* on the item *j*. When estimating the specific parameters of *GE* factors, the problem of data sparsity also needs to be considered. Therefore, it is necessary to compress the parameters in which the equation for compressing parameters is shown in Equation (14).

$$\theta_u^{(t)} = \frac{|I_u|}{|I_u| + \lambda} * \frac{\sum_{j \in p_u} R_{u,j}^{(t)} x_{u,j}}{\sum_{j \in p_u} x_{u,j}^2} \tag{14}$$

Where, $\theta_u^{(t)}$ refers to the *t*th parameter based on users and I_u denotes the set of all items scored by the user u. Moreover, $x_{u,j}$ represents the explanatory variables of the user *u* and the *j*th item. For example, $Time(user)^{1/2}$ can be calculated by using Equation (15):

$$x_{u,j} = Time(user)^{1/2} = \sqrt{t_{u,j} - t_u^{first}} \tag{15}$$

Where, $t_{u,j}$ refers to the time when the user *u* scores the item *j* and t_u^{first} represents the first scoring time of the user *u* scores for items. In order to make predicted result more accurate, *GE* is taken into account in the algorithm. Based on the aforementioned equation, *KNN* algorithm considering *GE* is improved, as shown in Equation (16).

$$\hat{R}_{u,j} = om + \sum_{i=1}^{t} \theta_i x_{u,j} + \frac{\sum_{n \in N(u,j)} \overline{sim(j,n)} (R_{u,n} - om - \sum_{i=1}^{t} \theta_i x_{u,n})}{\sum_{n \in N(u,j)} \overline{sim(j,n)}} \tag{16}$$

Where, *om*, that is Overall mean, refers to the mean score of all items. Moreover, *t*, θ_i and $x_{u,j}$ represent the number of *GE*, the *i*th *GE* and the explanatory variables of the user *u* and the *j*th item. Moreover, *N*(*u*, *j*) represents the set of k items most similar to the item *j* among all items rated by the user *u* and $\overline{sim(j,n)}$ denotes the similarity between compressed items *j* and *n*.

5. Hybrid recommendation algorithm and evaluation on recommendation performance

5.1. Design of hybrid recommendation algorithm

Owing to each recommender system shows its own advantages and disadvantages, in the process of recommendation, each recommender system will give different recommendation results for a specific user. How to improve the overall performance of the recommender system by effectively utilizing the recommendation results of different recommender systems? The purpose of hybrid recommendation is to develop advantages while avoiding disadvantages by combining different recommendation algorithms, thus making the recommendation result conform to users' demand.

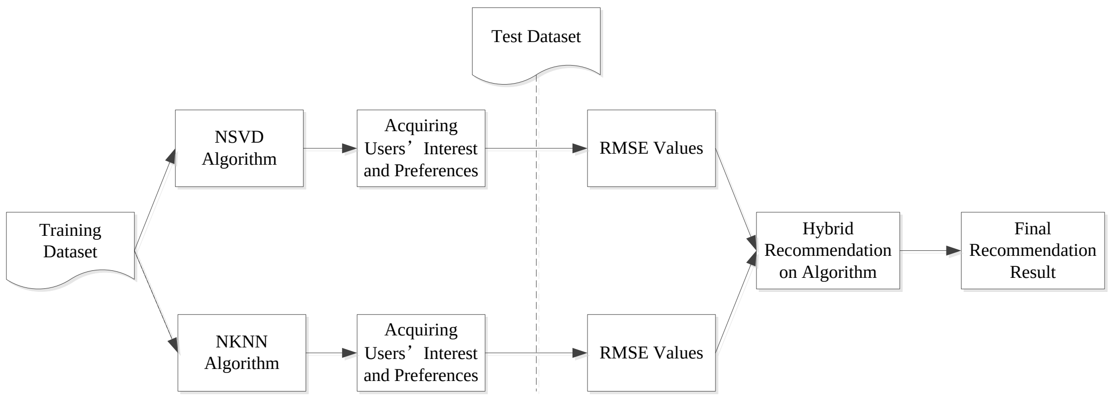

Test Dataset
NSVD Algorithm
Acquiring Users' Interest and Preferences
RMSE Values
Training Dataset
Hybrid Recommendation on Algorithm
Final Recommendation Result
NKNN Algorithm
Acquiring Users' Interest and Preferences
RMSE Values

Fig. 1. Design framework of the recommender system

In Fig.1, we acquired the users' interest and preferences by *NSVD* and *NKNN* (New K-nearest neighbor) algorithm separately. During the task of predicting scores, all scores obtained when *NSVD* and *NKNN* have the optimal performances (the lowest root mean squared error (*RMSE*)) are attained at first. Owing to the dataset of *NSVD* is the same as that of *NKNN* algorithm, the means of *NSVD* and *NKNN* algorithms are calculated by using the hybrid recommender system in view of the predicted score of a specific user for a certain item. Finally, the mean is taken as the predicted score.

5.2. Evaluation on recommendation performance

The quality of recommendation result of a recommender system needs to be evaluated. Recommendation accuracy is one of the most commonly used indices for recommender systems. However, the method for measuring accuracy varies in different steps of recommender systems. When predicting scores of items, three indices (including mean absolute error (*MAE*), *RMSE* and normalized mean absolute error (*NMAE*)) are generally applied [2]. The calculation equations are shown as follows:

$$MAE=\frac{1}{n}\sum_{i=1}^{n}\left|R_{u,i}-R_{u,i}^{(real)}\right| \tag{17}$$

$$RMSE=\sqrt{\frac{1}{n_u}\sum\left|R_{u,i}-R_{u,i}^{(real)}\right|^2} \tag{18}$$

$$NMAE = \frac{MAE}{R_{\max} - R_{\min}} \tag{19}$$

In the above equations, n denotes the number of items in the system scored by the user u and $R_{u,j}$ refers to the predicted score of the user u on non-scored item i attained through score prediction. Additionally, $R_{u,i}^{(real)}$ refers to the actual score (real value) of the user u on the item i in test dataset and n_u denotes the number of user-item pairs in the recommender system. Moreover, R_{max} and R_{min} denote the maximum and minimum scoring intervals of users, respectively.

6. Analysis on experimental data and results

6.1. Experimental data

The MovieLens dataset (www.movielens.org) was applied. MovieLens is a well-known movie scoring website and an early experimental recommender system. The dataset consists of three datasets with different sizes:

ML-100K: it is the smallest dataset in MovieLens, with a total of 100,000 scoring records, which are obtained from the scoring of 943 users on 1,682 movies. Moreover, the scoring time of users is shown. Each user scored 20 movies at least and records with fewer than 20 movies scored by users have been deleted.

ML-1M: it is a large dataset in MovieLens, with 1,000,209 scoring records, which are generated by 6,040 users scoring 3,900 movies. In the dataset, the scoring time of each scoring record is given. At the same time, records with fewer than 20 movies scored by users have been deleted from the dataset.

ML-10M: it is a large-scale dataset in MovieLens, with 10,000,054 scoring records, and 10,681 movies were scored by 71,567 users. The dataset contains time information for scoring. Moreover, this dataset also provides 95,580 label data of the users on the movies.

The genres were regarded as the factors of movies, those genres were consisted by 19 different themes such as comedy, romance, fantasy, drama, adventure, animation and so on. In the experiment, 80% of the data from each dataset were taken as the training set and 20% of the data was considered as the test dataset.

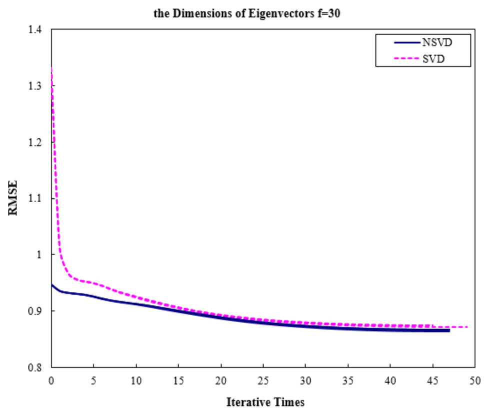
6.2. Experimental results and analysis on score prediction

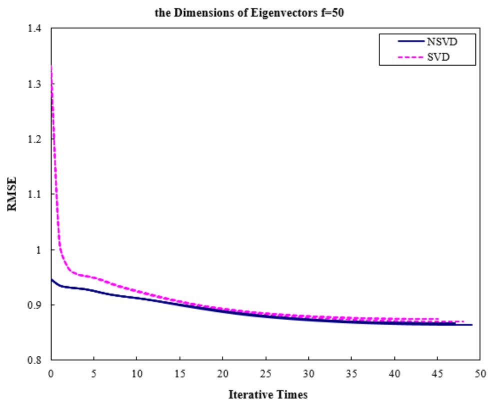
6.2.1. Experimental results and analysis based on NSVD algorithm

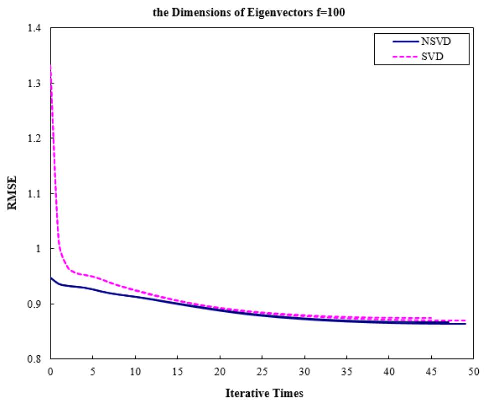
Experiments were carried out on the three datasets in MovieLens with different sizes by applying *NSVD* recommendation algorithm, *KNN* algorithm and hybrid algorithm. In terms of predicting scores of items, the accuracy of the predicted result was evaluated by using *RMSE*.

Aiming at different dimensions of eigenvectors, Fig.2 shows the prediction accuracies of *SVD* and *NSVD* algorithms for the large dataset ML-1M in MovieLens. Moreover, the dimensions of 20, 30, 50 and 100 were separately selected for the eigenvectors.

It can be seen from Fig. 2 that with the growth of iterative times, the performances of *SVD* and *NSVD* algorithms progressively rose until they converged and the effect of the initial iteration was more significant. Especially for *SVD* algorithm, its prediction performance significantly improved in the first two iterations in which *RMSE* values reduced by about 0.3 after the first two iterations. The performances of the two algorithms slightly increased with the growth of dimensions of eigenvectors. Additionally, the time for iteration multiplied while the effect of prediction scoring was insignificantly improved. If the dimensions of eigenvectors are 20, 30, 50 and 100, the prediction errors (*RMSE*) of *SVD* algorithm were 0.8747, 0.8722, 0.870 and 0.8697 while those of *NSVD* algorithm were 0.8668, 0.8648, 0.8636, and 0.8629, respectively. With the different dimensions of eigenvectors, the prediction errors (*RMSE*) of the two algorithms reduced by 0.005 and 0.0039, respectively. On the condition of having the same dimension of eigenvectors, the prediction error of *NSVD* algorithm was greatly lower than that of *SVD* algorithm. The *RMSE* value of *NSVD* algorithm reduced by 0.0074 on average compared with that of *SVD* algorithm while the performance of scoring prediction improved by 0.84%.







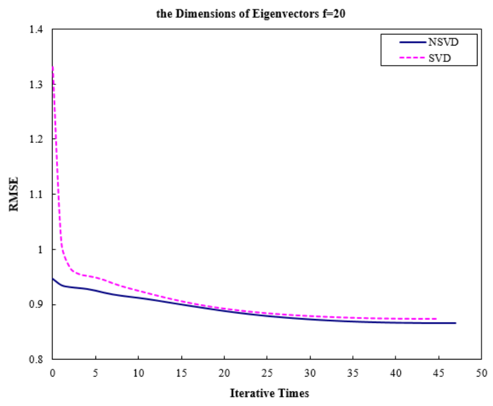


Fig.2. Change curves of recommendation performances of SVD and NSVD algorithms with iterative times base on the dataset ML-1M in MovieLens

Related experiments were also carried out on the other two largest and smallest datasets in MovieLens, ML-10M and ML-100K, respectively. The test results showed that the performance of *NSVD* algorithm was greatly superior to that of *SVD*. The performance of the recommender system in scoring prediction insignificantly changed with changing dimension of eigenvectors. Therefore, in subsequent research, the experiment was conducted only when the dimension of eigenvector is 50. In Fig. 3, the minimum prediction errors obtained by using *NSVD* and *SVD* algorithms for different datasets are shown.

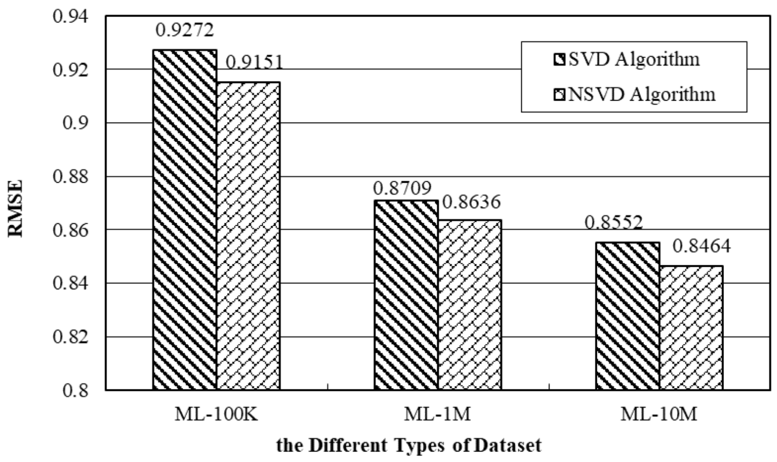


Fig. 3. Changes of RMSEs of SVD and NSVD algorithms in different datasets

In Fig. 3, the performances of *SVD* and *NSVD* algorithms slightly increased with the growth of data in datasets. When the size of the dataset rose from 100 k to 1 M, the *RMSEs* of *SVD* and *NSVD* algorithms decreased by 0.0563 and 0.0515, respectively. When the size of the dataset rose from 1 M to 10 M, the RMSEs of SVD and NSVD algorithms reduced by 0.0157 and 0.0172, respectively. It can be seen that the performance of scoring prediction greatly improved with increasing size of datasets in the initial stage. However, with the further growth of the size of dataset, the performance insignificantly increased, approximating to a stable state. Overall, when the dimension of eigenvectors was

50, the *NSVD* algorithm showed the optimal performance of scoring prediction (that is, the lowest *RMSE* of 0.8464) in the dataset of ML-10M.

6.2.2. Experimental result based on NKNN algorithm and analysis

By separately utilizing *KNN* and *NKNN* algorithms, experiments were conducted on different datasets. During the experiment, the values of k were 5, 10 and 15 and the specific experimental result is shown in Table 2.

Table 2

Performances of KNN and NKNN algorithms in scoring prediction on different datasets

	ML*-100*K			***ML*-1*M***			***ML*-10*M***		
K	**5**	**10**	**15**	**5**	**10**	**15**	**5**	**10**	**15**
KNN	1.0789	1.0758	1.0714	1.0238	1.0155	1.0164	1.0206	1.0169	1.0142
NKNN	0.9683	**0.9462**	0.9509	0.9084	**0.8827**	0.8913	0.8965	**0.8776**	0.8794

It can be seen from Table 1 that during the experiment on different datasets, the performance of *NKNN* algorithm was greatly improved compared with *KNN* algorithm. The *RMSE* decreased by about 0.13 on average and the improvement of the overall performance reached 12.13% on average. During experiments on different datasets, on the condition of k=10, *NKNN* algorithm showed a favorable performance. Moreover, on the condition of K=10 and the dataset of experiment was ML-10M, *NKNN* algorithm exhibited the optimal *RMSE* of 0.8776. Through the whole experiment, it can be seen that the data sparsity showed a greater impact on the coarser model. The *GE* factor has a more significant impact because the prediction result obtained through *KNN* algorithm was the weighted average of similarity and users' rating. The more similarity-independent factors (i.e. *GE* factors) are included in user ratings, the less satisfactory the final results are.

6.2.3. Experimental result and analysis on scoring prediction by using hybrid recommender system

The means of predicted scores of various movies obtained by using *NSVD* and *NKNN* algorithms were calculated in the experiment on the hybrid algorithm. In *NSVD* algorithm, the predicted scores of various movies when the dimension of eigenvector was 50 and *RMSE* (0.863593) was the lowest were selected. In *NKNN* algorithm, the predicted scores of movies under *K*=10 were mainly selected. The experimental result is shown in Fig. 4.

In Fig. 4, the performances of various recommendation algorithms slightly improved with growing size of datasets. The data sparsity slightly reduced with the growth of dataset size, causing the reduction of *RMSE*.

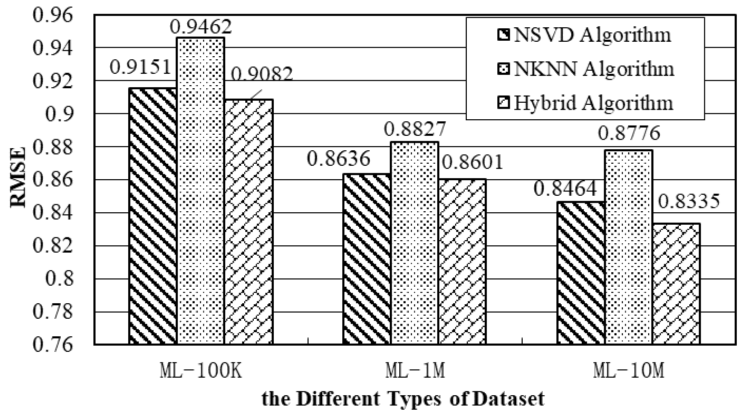


Fig. 4. A comparison of performances of various recommendation algorithm

The overall effect of scoring prediction by using *NSVD* algorithm was greatly improved compared with *KNN* algorithm in which *RMSE* averagely decreased by about 0.03. The performance of hybrid recommendation algorithm slightly improved based on *NSVD* and *NKNN* algorithms in which *RMSE* reduced by about 0.01 on average. It was mainly because the hybrid algorithm combines the local advantages of *NSVD* and *KNN* to make the scoring effect more obvious.

7. Conclusions and prospects

In our study, we presented the following contributions over the current state-of-the-art: (i) An improved version of SVD (NSVD) that considering the characterization factors of users were proposed, and the parameters were optimized by stochastic gradient descent (*SGD*) algorithm. (ii) In order to enhance the reliability of the predicted results, we improved the method of similarity of KNN, and global effects factors were considered in the NKNN algorithm. (iii) A hybrid algorithm was proposed by the NSVD and NKNN algorithms achieved a favorable effect. (iv) The performances of our approaches were greatly reduced in a cold start scenario (either users or items with few ratings).

However, various problems including the diversity of recommended items and the data sparsity still need to be further solved. The subsequent research can be carried out from the following aspects: (i) Data sparsity; although the solution to data sparsity is involved in the algorithm, there are still some shortcomings in the existing method with the further increase of the number of recommended items in the future. (ii) Method for acquiring users' interest and preference; other information about users themselves, such as gender, age, time of scoring items and so on, needs to be taken into account in future study. (iii) Acquisition of interactive information between users and items; in addition to scoring, interactive information between users and items includes users' browsing information and information on non-scored items which have been purchased. Certainly, there are many very useful text information related to users' evaluation. If the information can be favorably analyzed, it will also play a positive role in recommendation performance. (iv) Local calculation of data; the information of users and items is dynamically changed. As a result, if the correlation between users and items needs to be re-calculated during each change, there will be a huge calculation amount. It is suggested to design a dynamic algorithm to acquire final calculation result only by calculating the changed part.

Acknowledgement

This work was supported by the Key Program of the Foundation for Young Talents in the Colleges of Anhui Province under Grant 2013SQL097ZD; and the Program of Quality Engineering in the Colleges of Anhui Province under Grant 2017sjjd101, 2017jyxm0965, 2018xfsyxm010; and the Foundation of Anhui Open University under Grant No. JY19-01.

R E F E R E N C E S

[1]. *A. Singhal, P. Sinha, R. Pant.* Use of deep learning in modern recommendation system: A summary of recent works. arXiv preprint arXiv:1712.07525, 2017.

[2]. *R. Baeza-Yates, B.A.N. Ribeiro.* Modern information retrieval. New York: ACM Press; Harlow, England: Addison-Wesley, 2011.

[3]. *M. Balabanović, Y. Shoham.* Fab: content-based, collaborative recommendation. Communication of the ACM, **vol. 40**, no. 3, 1997, pp. 66-72.

[4]. *M. Degemmis, P. Lops, G. Semeraro*. A content-collaborative recommender that exploits WordNet-based user profiles for neighborhood formation. User Modeling and User-Adapted Interaction, **vol. 17,** no. 3, 2007, pp. 217-255.

[5]. *T.T.S. Nguyen, H.Y. Lu, J. Lu.* Web-page recommendation based on web usage and domain knowledge. IEEE Transactions on Knowledge and Data Engineering, **vol. 26,** no.10, 2014, pp. 2574-2587.

[6]. *S. Langer, J. Beel.* Apache lucene as content-based-filtering recommender system: 3 Lessons Learned. arXiv preprint arXiv:1703.08855, 2017.

[7]. *Y. Deldjoo, M. Elahi, P. Cremonesi, et al.* Content-based video recommendation system based on stylistic visual features. Journal on Data Semantics, **vol. 5**, no. 2, 2016, pp. 99-113.

[8]. *P. Resnick, N. Iacovou, M. Suchak, et al.* An open architecture for collaborative filtering of netnews. In Proceedings of the 1994 ACM Conference on Computer Supported Cooperative Work (CSCW), 1994, pp. 175-186.

[9]. *G. Linden, B. Smith, J. York.* Amazon.com recommendations: Item-to-item collaborative filtering. IEEE Internet computing, **vol 1**, 2003, pp. 76-80.

[10]. *Z. Yang, B. Wu, K. Zheng, et al.* A survey of collaborative filtering-based recommender systems for mobile internet applications. IEEE Access, **vol. 4**, 2016, pp. 3273-3287.

[11]. *S. Baluja, R. Seth, D. Sivakumar, et al.* Video suggestion and discovery for youtube: taking random walks through the view graph. In Proceedings of the 17th international conference on World Wide Web. ACM, 2008, pp. 895-904.

[12]. *M. Nilashi, O. B. Ibrahim, N. Ithnin*. Multi-criteria collaborative filtering with high accuracy using higher order singular value decomposition and Neuro-Fuzzy system. Knowledge-

Based Systems, **vol. 60**, no. 2, 2014, pp. 82-101.

[13]. *D. A. Adeniyi, Z. Wei, Y. Yongquan*. Automated web usage data mining and recommendation system using K-Nearest Neighbor (KNN) classification method. Applied Computing and Informatics, **vol. 12**, no. 1, 2016, pp. 90-108.

[14]. *W. Jian, J. He, C. Kai, et al.* Collaborative filtering and deep learning-based recommendation system for cold start items. Expert Systems with Applications, **vol. 69**, 2017, pp. 29-39.

[15]. *N.E.I. Karabadji, S. Beldjoudi, H. Seridi, et al.* Improving memory-based user collaborative filtering with evolutionary multi-objective optimization. Expert Systems with Applications, **vol. 98**, 2018, pp. 153-165.

[16]. *M. Jamali, M. Ester.* TrustWalker: a random walk model for combining trust-based and item-based recommendation. In Proceedings of the 15th ACM SIGKDD international conference on Knowledge discovery and data mining. ACM, 2009, pp. 397-406.

[17]. *M. Reshma, R. R. Pillai.* Semantic based trust recommendation system for social networks using virtual groups[C]//2016 international conference on next generation intelligent systems (ICNGIS). IEEE, 2016, pp. 1-6.

[18]. *L. Ravi, S. Vairavasundaram.* A collaborative location-based travel recommendation system through enhanced rating prediction for the group of users. Computational intelligence and neuroscience, **vol. 2016**, 2016, pp.1-28.

[19]. *B. Yang, Y. Lei, J. Liu, et al.* Social collaborative filtering by trust. IEEE Transactions on Pattern Analysis and Machine Intelligence, **vol. 39**, no. 8, 2017, pp. 1633-1647.

[20]. *R. M. Bell, Y. Koren.* Scalable collaborative filtering with jointly derived neighborhood interpolation weights. In icdm. IEEE, 2007, pp. 43-52.